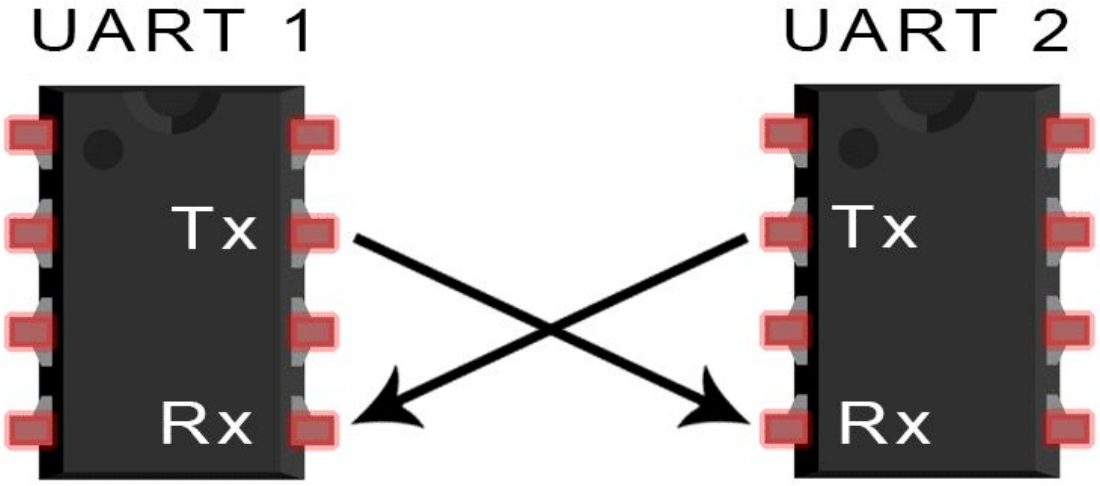
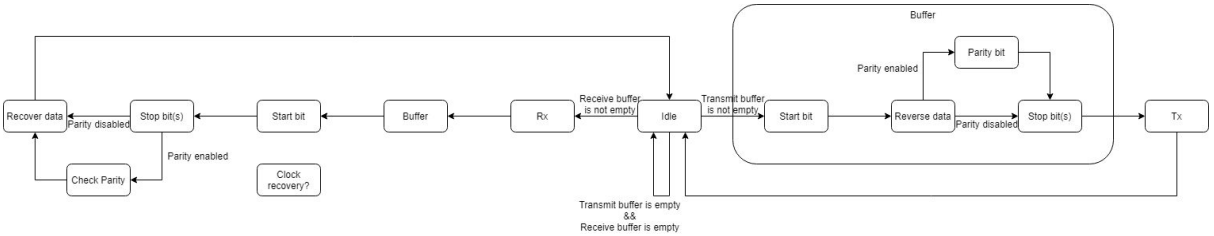
Report UART ontwerpen & bouwen



Door: Stefan Hobeijn & Daan Knippenberg
Klas: T42
Vak: DES
Docent: Martin Ederveen
Datum: 25-3-2020

Design

1e versie

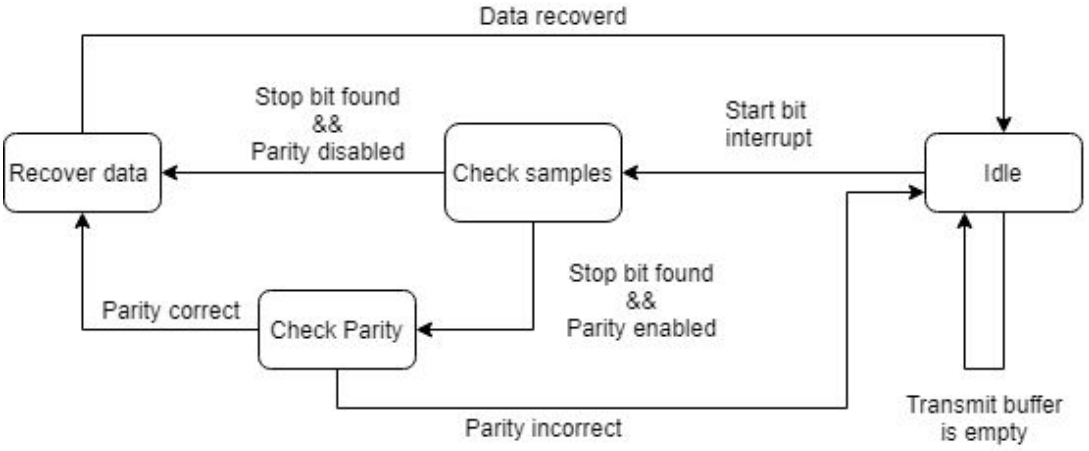


Feedback van versie 1

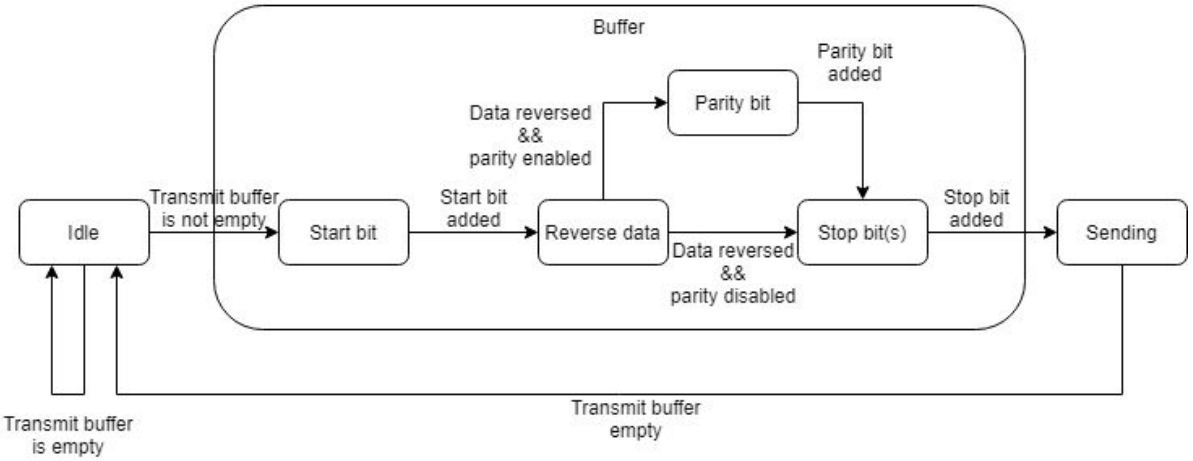
- bij elke overgang moet een conditie bijkomen
- de termen in de state zelf geven te weinig informatie
- hoe weet de receiver wanneer hij moet gaan lezen?, door een interrupt te gebruiken
- meer in de stijl van een state diagram, nu nog in combinatie met een flowchart
- iets met samples toevoegen en controleren

2e versie

Hieronder is het receive gedeelte van de uart te zien. We hebben het voor de leesbaarheid even in 2 afbeeldingen geknipt, dus je ziet 2x idle, maar die zijn dus hetzelfde.



Hieronder is het transmit gedeelte van de uart te zien.



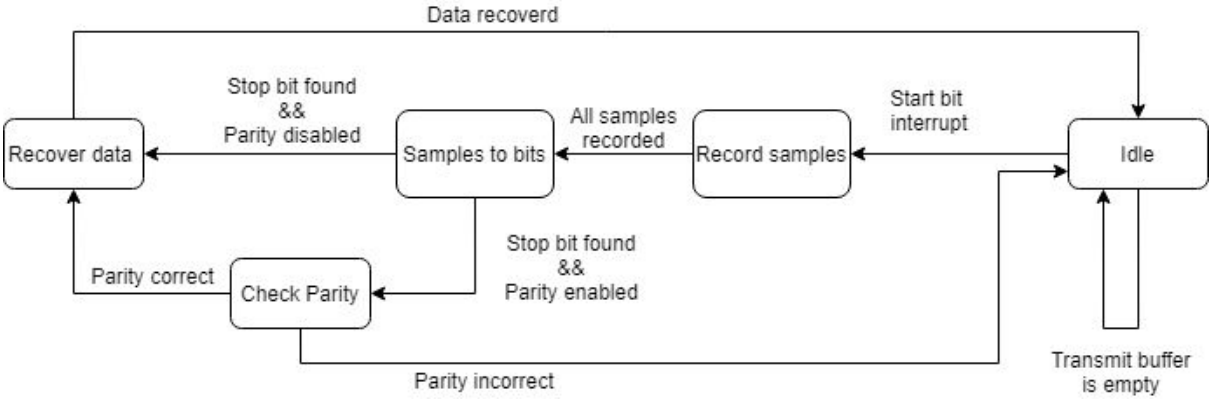
Feedback van versie 1

- RX: parity kan je pas controleren nadat op een of andere manier de bits uit de samples zijn gehaald, voorstel zou zijn om check samples in tweeen te splitsen, namelijk record samples & samples2Bits
- TX is prima

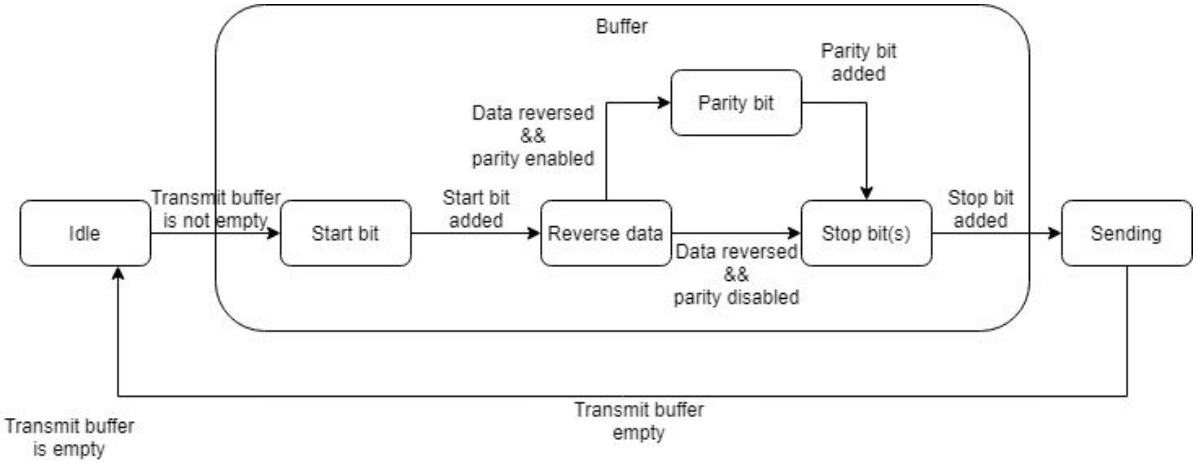
3e versie

In deze versie hebben wij de feedback van de docent toegepast en de feedback lag eigenlijk alleen op het receive gedeelte.

Hieronder is het receive gedeelte van de uart te zien. We hebben het voor de leesbaarheid weer in 2 afbeeldingen geknipt, dus je ziet 2x idle, maar die zijn dus hetzelfde.



Hieronder is het transmit gedeelte van de uart te zien.



Testplan

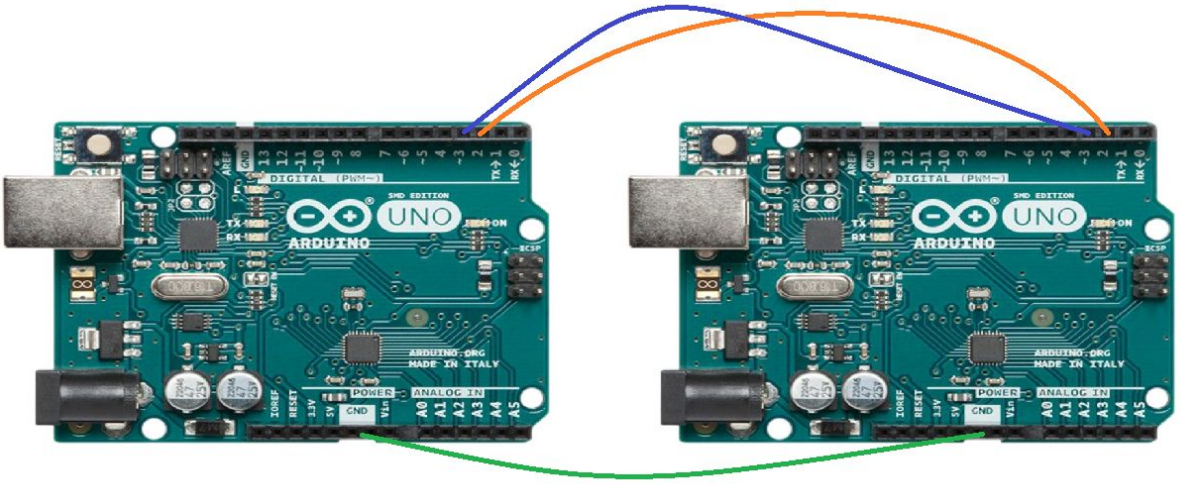
Tussendoor te testen

Om onze UART te testen gaan wij een usb to ttl gebruiken.

Wij gebruiken PuTTY om de output van ons programma uit te lezen, en om de input voor ons programma in te typen.

Eindtest

Uiteindelijk willen we dus 2 verschillende apparaten aan elkaar hangen en die met elkaar laten communiceren en dat is de laatste test, want daar draait de opdracht om.

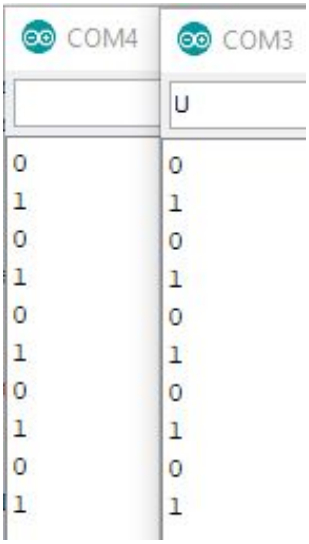


De transmitter heeft de RX pin op 3 zitten en de TX pin op 2 en bij de receiver is dit juist andersom, dus de RX pin zit op 2 en de TX pin op 3.

Testrapport

Test V1

Wij hebben onze eerste versie getest. Dit bevat het sturen en ontvangen van individuele bits. De transmitter(COM3) maakt een bericht van een input. Dit is nu nog maar een karakter. Vervolgens wordt er een stopbit en een startbit toegevoegd. Dit bericht word verstuurd met een baudrate van 1.

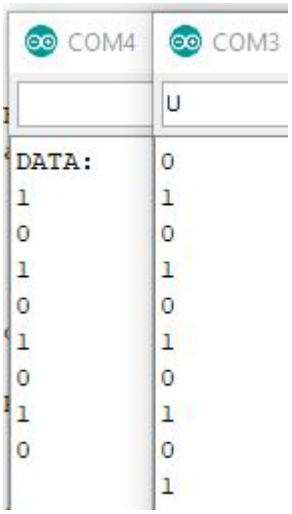


COM 3 is de transmitter (Arduino).
COM 4 is de receiver (ESP32).

In de bovenstaande afbeelding zie je wat de input is en wat er wordt verstuurd door de transmitter. Ook zie je wat er wordt ontvangen door de receiver.
Ook kun je zien dat de least significant bit als eerst wordt verstuurd wanneer je de stop en de startbit weghaalt, aangezien een U in binair 0101 0101 is.

Test V2

Wij hebben onze tweede versie getest. Wat nieuw is aan deze versie is dat de start en de top bits uit het bericht worden gefilterd.

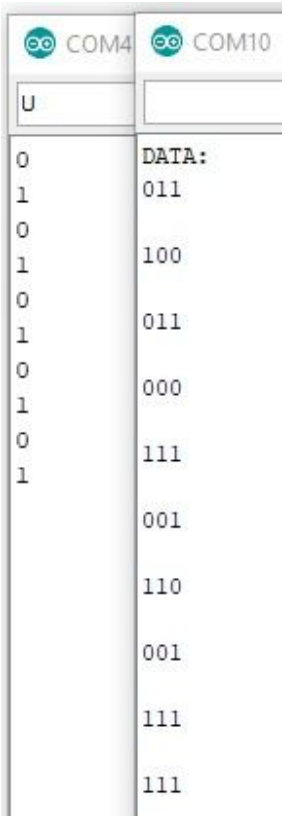


COM 3 is de transmitter (Arduino).
COM 4 is de receiver (ESP32).

In de bovenstaande afbeelding zie je dat de eerste bit en de laatste bit niet worden geprint door de receiver.

Test V3

Voor de test van de derde versie wilden wij met samples gaan werken. Hiervoor hebben wij drie bits naast elkaar staan. Elke rij van bits in de receiver is één verstuurde bit van de transmitter.
Verder is er een timer interrupt geïmplementeerd.

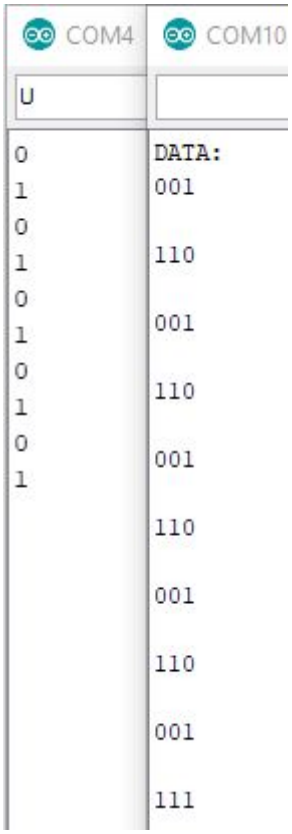


COM 4 is de transmitter (Arduino).
COM 10 is de receiver (Arduino).

in de bovenstaande afbeelding zie je dat de ontvangen data niet klopt. Er staan teveel fouten in de ontvangen bits dat het ontvangen karakter niet meer uit te lezen is.

Test V4

Voor de vierde test hebben we het interval van de timer aangepast. Deze hebben we aangepast d.m.v. een formule. Hierdoor hebben we max maar een fout in de data staan, en soms helemaal geen.



COM 4 is de transmitter (Arduino).
COM 10 is de receiver (Arduino).

in de bovenstaande afbeelding zie je dat de ontvangen data bijna overeenkomt met de verstuurde data. Uit deze samples is de juiste data weer te halen.

Formule

We hadden wat problemen om de interrupt timer goed in te stellen en zijn toen de onderstaande formule tegengekomen, waar je de frequentie die je wilt kunt invullen en voor de timer clock frequentie moet je de snelheid van de arduino zelf invullen.

One small improvement. The top value should 1599 instead of 1600.
The formula is:

Code: [Select]

```
Target Timer Count = (1 / Target Frequency) / (1 / Timer Clock Frequency) - 1
```

With value 1599 the timing is very accurate..

Op de onderstaande formule is hij ingesteld op 9600 baudrate

$$\frac{(1 / 9600)}{(1 / 16000000)} - 1 = 1665.666667$$

$$\frac{(1 / (9600 * 3))}{(1 / 16000000)} - 1 = 554.5555565$$

Onze UART is ook werkend op 10 keer zo snel, zie onderstaande formule (op 96000 baudrate)

$$\frac{(1 / 96000)}{(1 / 16000000)} - 1 = 159$$

$$\frac{(1 / (96000 * 3))}{(1 / 16000000)} - 1 = 52.33$$

Uitbreidingen

Onze opdracht kan op de volgende gebieden nog worden uitgebreid:

- Baudrate verhogen (is nu 1 seconde)
- Code verbeteren, qua methodes en leesbaarheid
- Parity toevoegen
- Dat de gebruiker 1 of 2 stop bits kan selecteren
- Dat de gebruiker zelf een baudrate kan selecteren